MANUAL DE USUARIO

XBOARD RELAY

MCI-MA-1413 | REV. 1.0

Ingeniería MCI Ltda.

Luis Thayer Ojeda 0115 of. 1105, Providencia, Santiago, Chile.

+56 2 23339579 | www.olimex.cl | info@olimex.cl

Ingeniería MCI Ltda.

Luis Thayer Ojeda 0115 Oficina 1105
Providencia, Santiago, Chile

www.olimex.cl
info@olimex.cl

Tel: +56 2 23339579
Fax: +56 2 23350589

® MCI Ltda. 2014

Atención: *cambios y modificaciones hechas en el dispositivo, no autorizados expresamente por MCI, anularán su garantía.*

Código Manual: MCI–MA-1413

CONTENIDO

CONTENIDO 3
INTRODUCCIÓN 4
CARACTERÍSTICAS GENERALES 4
INTERFAZ DE USUARIO / PARTES DEL DISPOSITIVO 4
PROGRAMAS Y LIBRERÍAS A UTILIZAR 5
CONECTANDO POR PRIMERA VEZ XBOARD RELAY. 7
CONFIGURACIÓN DEL DISPOSITIVO. 8
PROGRAMACIÓN DE LA TARJETA. 12
CONTROL A TRAVÉS DE APLICACIÓN SOULISS 17
CARACTERÍSTICAS ELÉCTRICAS 18
CARACTERÍSTICAS MECÁNICAS 18
HISTORIA DEL DOCUMENTO 23

INTRODUCCIÓN

Con la tarjeta Xboard Relay se puede monitorear datos a través de internet y también controlarlos a través del mismo. Esta tarjeta combina un procesador Atmega 32u4 y un chip wiz5100 que es completamente compatible con el Arduino Leonardo y las librerías Souliss para acceso a internet. Además posee un socket XBee incorporado para conexiones inalámbricas y 2 relés que facilitan el monitoreo y control a través de internet.

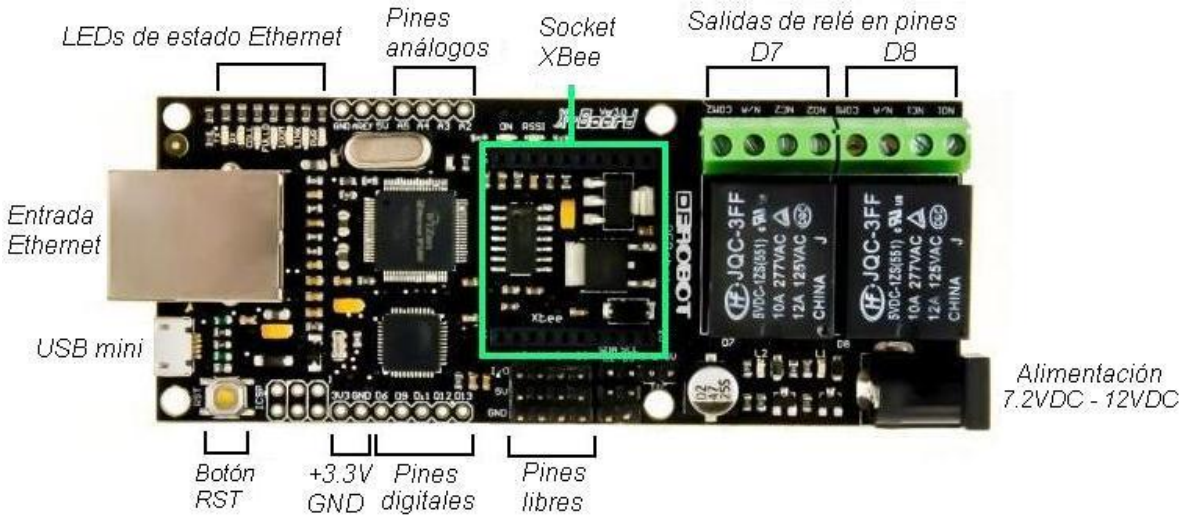
Esta tarjeta es ideal para controlar una gran cantidad de artefactos y equipos eléctricos o electrónicos a distancia. Este control se puede realizar desde internet, (aplicación Souliss para Android e IOS), con la activación de un pulsador o automatización a través de sensores. No se requiere de un programador externo para programarlo. Un cable micro-USB es el único hardware que se necesita para cargar un sketch.

CARACTERÍSTICAS GENERALES

- Microcontrolador ATMega 32u4
- Chip Wiz5100 (compatibilidad para protocolo TCP/IP).
- Compatibilidad Arduino.
- Socket XBee.
- 15 Pines para uso del usuario entre los cuales se encuentran:
    - I/O digitales: D2, D3, D4, D5, D6, D9, D11, D12, D13.
    - Análogas: A0, A1, A2, A3, A4, A5.
- 2 Relés incorporados en salidas digitales: D7, D8.
- Jack DC alimentación externa: 7,2 VDC a 12 VDC.
- Botón de reset.
- Entrada Ethernet con conector tipo RJ45
- Base acrílica para montar Xboard Relay.

INTERFAZ DE USUARIO / PARTES DEL DISPOSITIVO

A continuación se presentan las partes que componen la tarjeta Xboard Relay con sus respectivas descripciones:



- Entrada Ethernet: Conector tipo RJ-45, para conexión a la red LAN.
- USB mini: Entrada de USB para cargar sketch a Arduino y brindar alimentación.
- LEDs de estado Ethernet: LEDs de estado Link, 100/10mb, full dúplex, TX, RX, Col y Power.
- Alimentación 7.2VDC – 12VDC: Conector de alimentación a la tarjeta de manera externa.
- Salidas de relé: Terminal block conectados a los relés para el control de artefactos de un consumo y voltaje alto (220VAC).
- Socket XBee: Conector para montar módulos XBee o compatibles y crear sistema inalámbrico.
- I/O digitales: Entradas y salidas digitales que tolerantes a 5VDC.
- Pines análogos: Entrada de señales análogas tolerantes a 5VDC.
- Pines libres: Pines análogos y digitales libres para uso del usuario de distinto propósito.
- Botón reset: Al pulsarlo la tarjeta comienza a ejecutarse desde el principio del código de programación (se usa principalmente frente a un error o bloqueo de la tarjeta)
- +3,3 VDC y GND: Pines para alimentar dispositivos de 3,3VDC en conjunto con GND.

PROGRAMAS Y LIBRERÍAS A UTILIZAR.

La tarjeta Xboard Relay es controlada por un Microcontrolador Atmega 32U4, el cual es reconocido por la computadora como un Arduino Leonardo. La programación en sketch es cargada en este dispositivo.

Para poder configurar y cargar sketch a la tarjeta Xboard Relay, se necesita descargar el programa de Arduino y las librerías de Souliss que se cargarán en el programa recién nombrado.

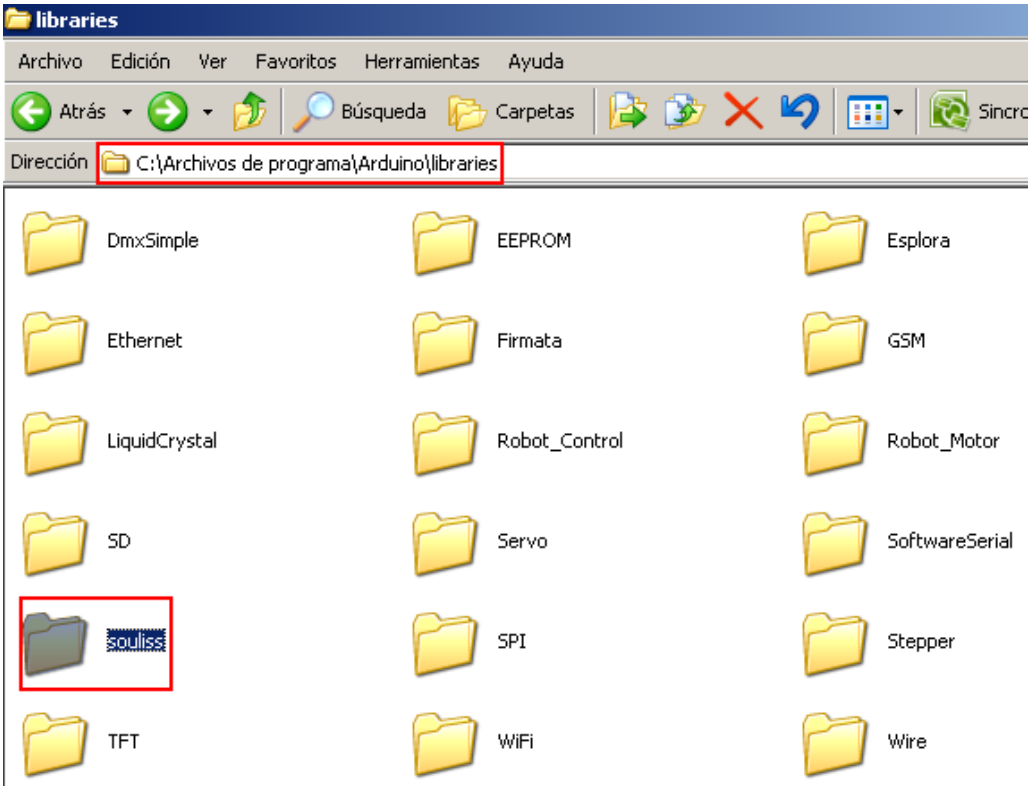
El programa Arduino utilizado para construir este manual y sus ejemplos es la versión Arduino 1.0.5-r2, y se puede descargar desde aquí.

Después de descargar e instalar el programa Arduino, es necesario descargar el programa o librerías de Souliss. La versión utilizada para este ejemplo es 1402 Souliss release A5.0.4, y se puede descargar aquí.

El entorno de programa Arduino detecta automáticamente estas librerías junto a sus ejemplos respectivos si es bien agregada a su carpeta “libraries”.

Para instalar las librerías se debe descomprimir el archivo .rar descargado, copiar la carpeta Souliss y pegarla en: Archivos de programa/Arduino/libraries.

La siguiente imagen muestra una correcta instalación de las librerías Souliss:



Nota: *Si se copia y pega la carpeta general descomprimida, se provocara un error en Arduino y no podrá ser usado. Debe ser copiada y pegada solo la carpeta Souliss. El nombre de esta carpeta no puede incluir espacios, puntos u otro carácter que no sea letras o números.*

CONECTANDO POR PRIMERA VEZ XBOARD RELAY.

La conexión principal para probar un correcto funcionamiento de la tarjeta Xboard Relay es conectar un cable USB mini a la computadora y seguido a la tarjeta. Cuando la tarjeta sea conectada el computador será reconocida como un Arduino Leonardo, es cuando se debe instalar sus drivers por defecto (con la instalación de Arduino se cargan automáticamente los drivers y no es necesario descargar ningún controlador externo).

Para cargar un sketch de prueba bastaría abrir el programa Arduino y seleccionar el ejemplo ubicado en el programa Arduino: Archivo/Ejemplos/01.Basics/Blink.

Arduino cargará un sketch en la tarjeta con un programa básico de parpadeo de LED, que en este caso se cambiará para utilizar como prueba uno de los relés de salida.

El cambio de pin para cambiar el LED por el relé, está en la línea de comando “int led = 13”, la cual se cambiara a “int led = 7”. Con esto se está diciendo a la tarjeta que utilice el pin D7 como salida.

Antes de cargar el sketch de prueba se debe seleccionar la tarjeta que se está utilizando, para esto se debe ir a: Herramientas/ Tarjetas/ Arduino Leonardo.

Para finalizar se debe seleccionar el puerto COM de la tarjeta Xboard Relay, que varía según el puerto USB al cual se conecte la tarjeta. Esta configuración se encuentra en: Herramientas/ Puerto Serial.

Si todo se encuentra en orden se deberá prender y apagar el primer relé conectado a D7.

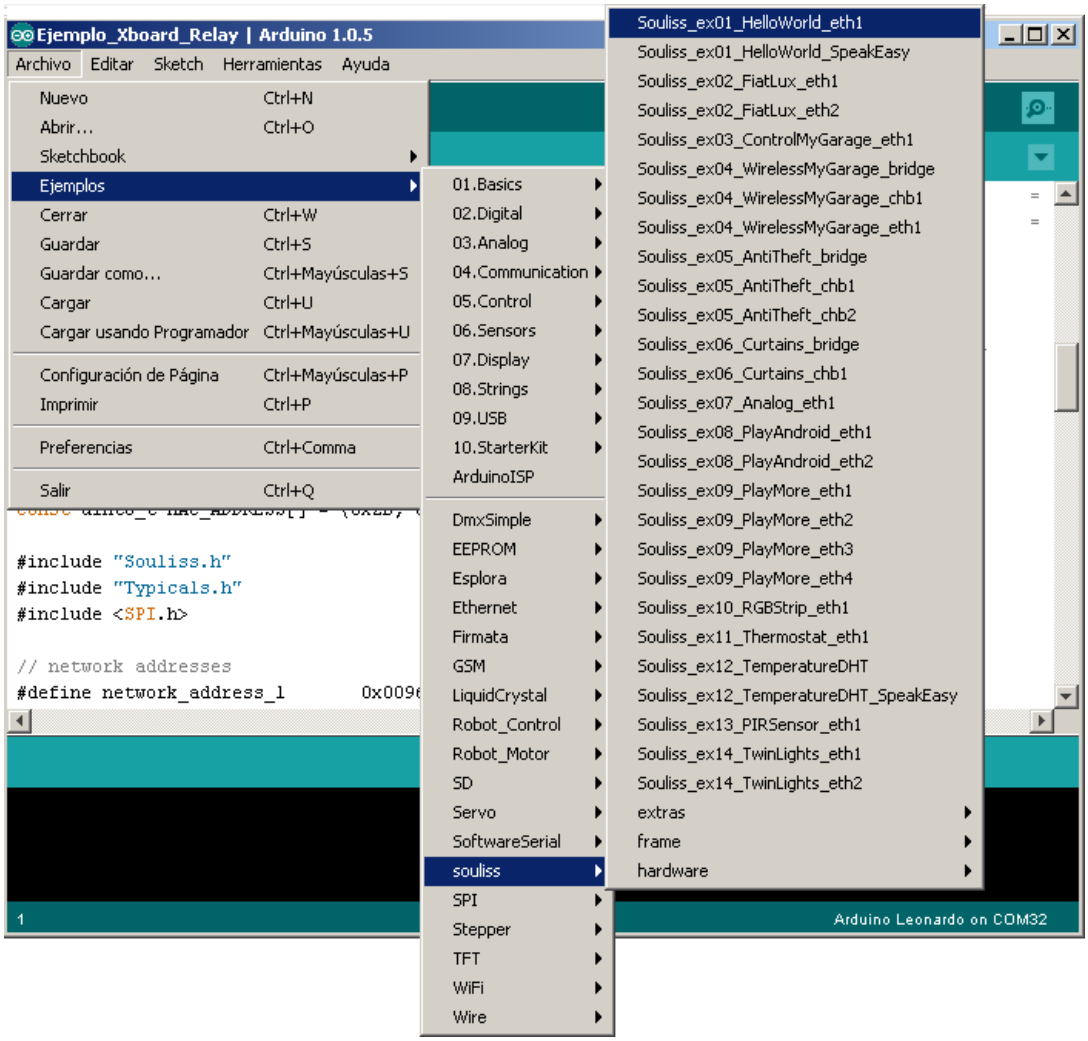
Nota: *Lo mismo se puede realizar para comprobar el buen funcionamiento del segundo relé conectado a D8.*

CONFIGURACIÓN DEL DISPOSITIVO.

En esta sección se explicará como configurar y programar la tarjeta Xboard Relay con un ejemplo básico, el cual se podrá manipular desde un celular con sistema operativo Android en conjunto a la aplicación Souliss.

Lo primero que se debe realizar es abrir el ejemplo Souliss_ex01_HelloWorld_eth1, el cual se ubica en la siguiente dirección: Archivo/ Ejemplos/ Souliss.

Guíese por la siguiente imagen para mayor comprensión:



Con el ejemplo ya cargado en el programa Arduino, se deberán cambiar una serie de líneas de programación y añadir unas cuantas más. Pero no solo basta con añadir o cambiar la

programación de Arduino, sino que también se debe cambiar la configuración de la librería de Souliss. Estas configuraciones son:

- IP (network_address_1).
- Puerta de enlace (GATEWAYTYPE).
- Tipo de tarjeta a utilizar (BOARDTYPE).

En este ejemplo se ha utilizado solo estas tres configuraciones. En programas con más requisito pueden ser cambiadas más variables como:

- QC_BOARDTYPE
- QC_GATEWAYTYPE
- MAC_ADDRESS = {};
- DYNAMICADDRESSING
- ETHERNETMACRAW
- DHT_SENSOR

A continuación se explicará dos maneras de cambiar estas configuraciones:

I. La primera, siendo la más complicada, se realiza cambiando las configuraciones directamente en el archivo "QuickCfg.h", que se encuentre en la siguiente dirección de carpeta: Archivos de programa/Arduino/ Souliss/conf.

Nota: Cada parámetro a cambiar tiene un explicativo de cómo activarlo y desactivarlo. Como referencia estos son los valores hexadecimales usados:

El valor hexadecimal 0x00 significa desactivado.
El valor hexadecimal 0x01 significa activado.

II. La segunda opción es más simple y rápida para el usuario. Se modifican las configuraciones de "QuickCfg.h" directamente desde el sketch, no modificándolo permanentemente, sino que se cambia la configuración para ese sketch de Arduino en específico.

Para cambiar una configuración directamente desde el sketch se debe anteponer a la línea de comando X_INSKETCH, donde X es el parámetro a cambiar. Por ejemplo, si se quiere cambiar la configuración de la tarjeta a ocupar directamente desde el sketch se debe escribir una definición de la siguiente manera:

```
#define  BOARDTYPE_INSKETCH
#define  QC_BOARDTYPE  0x30
```

En este caso se define a usar la tarjeta Xboard Relay la cual se identifica con el hexadecimal 0X30. Para configurar el resto de las posibles configuraciones existen estas definiciones:

- #define BOARDTYPE_INSKETCH
- #define GATEWAYTYPE_INSKETCH
- #define MAC_INSKETCH ( )
- #define DYNAMICADDRESSING_INSKETCH
- #define ETHERNETMACRAW_INSKETCH
- #define DHTSENSOR_INSKETCH

Nota: *Para que estas líneas de comando funcionen, se deben ingresar a Arduino antes de ingresar los comandos include como lo es por ejemplo: #include "Souliss.h"*

Quedando una programación como muestra la siguiente imagen:

Ejemplo_Xboard_Relay

```
/************************************************************************/

#define  BOARDTYPE_INSKETCH
#define  QC_BOARDTYPE  0x30        //Dispositivo a utilizar corresp

#define  GATEWAYTYPE_INSKETCH
#define  QC_GATEWAYTYPE  0x01    // Se define desactivado parámetro

#define  MAC_INSKETCH
const uint8_t MAC_ADDRESS[] = {0x2B, 0xA6, 0x49, 0x6B, 0x00, 0x01};

#include "Souliss.h"
#include "Typicals.h"
#include <SPI.h>

// network addresses
#define network_address_1       0x0096  // 0x0096 es igual a 150 de
#define network_my_subnet       0xFF00
#define network_my_supern       0x0000
```

Como serán valores que no cambien de un sketch a otro, algunos parámetros pueden ser agregados de forma permanente como lo son: address2 (se refiere a 192.168.X.0, donde X es address 2) y puerta de enlace a la cual estará conectada la tarjeta Xboard Relay. Estos valores se deben modificar en el archivo QuickCfg.h

En este caso, la puerta de enlace que genera el router es 192.168.0.2 y las direcciones IP que da por DHCP varían desde 192.168.0.100 hasta 192.168.0.190.

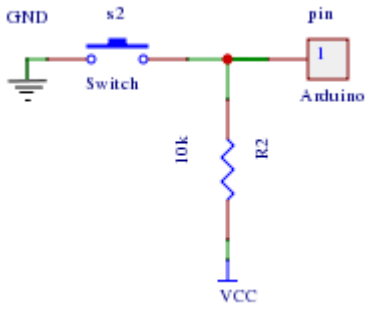
Al abrir el archivo QuickCfg.h se debe buscar las configuraciones de IP y modificarlas, quedando para este ejemplo de la siguiente manera:

```
#if(QC_ENABLE)
        uint8_t DEFAULT_BASEIPADDRESS[] = {192, 168, 0, 0};
        uint8_t DEFAULT_SUBMASK[] = {255, 255, 255, 0};
        uint8_t DEFAULT_GATEWAY[] = {192, 168, 0, 2};
#endif
```

Para finalizar la configuración, se explicara como agregar un botón pulsador que cambiara de estado el Relay7. Este botón pulsador se ha integrado para dar prueba del uso de una entrada digital a la tarjeta Xboard Relay.

Recordando, Relay7 está conectado a D7 y el botón pulsador estará conectado al pin digital D6, que posteriormente será programado como entrada.

Para crear este pequeño circuito se sugiere seguir el siguiente esquema electrónico:



Nota: *La IP queda con ultimo valor 0 (192.168.0.0) debido a que se puede cambiar este valor desde la programación del sketch de Arduino (este último valor IP también es llamado por programación address1).*

PROGRAMACIÓN DE LA TARJETA.

Para un usuario que desee ir directamente al código de programación para el sketch de Arduino, puede dirigirse al capítulo “Programa Sketch Arduino” ubicado al final de este manual.

Conociendo como configurar la tarjeta, la programación comienza agregando las definiciones para la configuración de tarjeta, MAC y activación de puerta de enlace.

Como ya se mencionó el código hexadecimal para la tarjeta Xboard Relay corresponde a 0X30, y la activación de puerta de enlace es mediante el código 0X01.

La dirección MAC, puede ser cambiada a opción del usuario, siempre y cuando esté bien construida.

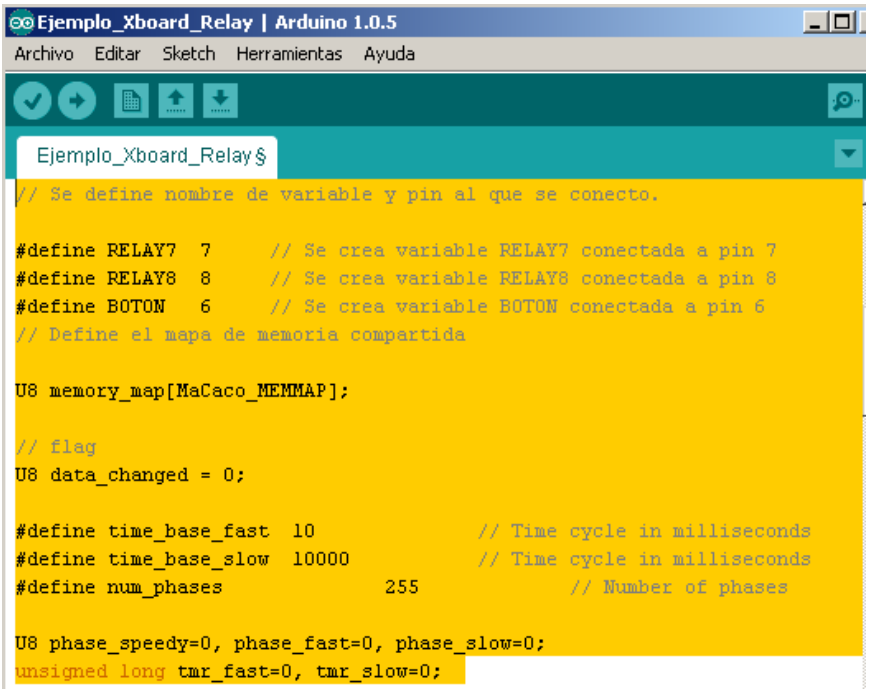
Luego se incluyen las librerías que Souliss utilizará en este ejemplo.

Para incluir una librería o archivo con programación se debe crear un #include.

Ejemplo_Xboard_Relay

```
/************************************************************************
#define  BOARDTYPE_INSKETCH
#define  QC_BOARDTYPE  0x30        //Dispositivo a utilizar corresp

#define  GATEWAYTYPE_INSKETCH
#define  QC_GATEWAYTYPE  0x01    // Se define desactivado parámetro

#define  MAC_INSKETCH
const uint8_t MAC_ADDRESS[] = {0x2B, 0xA6, 0x49, 0x6B, 0x00, 0x01};

#include "Souliss.h"
#include "Typicals.h"
#include <SPI.h>

// network addresses
#define network_address_1       0x0096  // 0x0096 es igual a 150 de
#define network_my_subnet       0xFF00
#define network_my_supern       0x0000
```

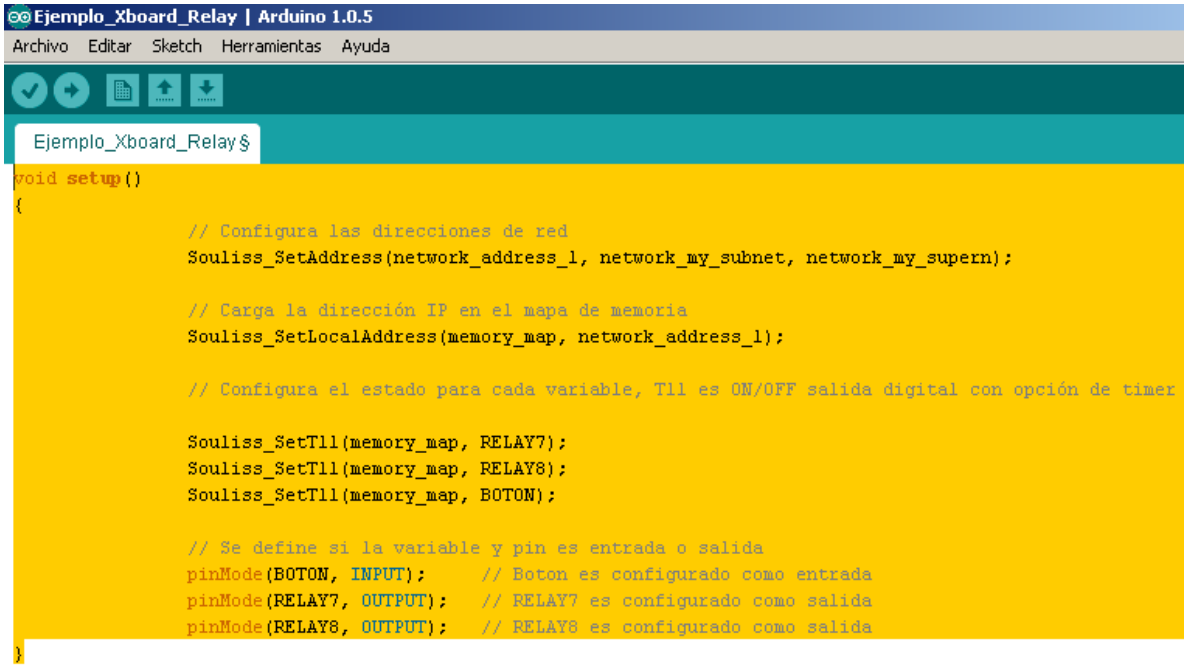
A continuación se ingresan las definiciones del último número IP que adquirirá la tarjeta, el cual será 150, construyendo una IP igual a 192.168.0.150.


Ejemplo_Xboard_Relay | Arduino 1.0.5
Archivo Editar Sketch Herramientas Ayuda
Ejemplo_Xboard_Relay §
// Se define nombre de variable y pin al que se conecto.
#define RELAY7 7 // Se crea variable RELAY7 conectada a pin 7
#define RELAY8 8 // Se crea variable RELAY8 conectada a pin 8
#define BOTON 6 // Se crea variable BOTON conectada a pin 6
// Define el mapa de memoria compartida
U8 memory_map[MaCaco_MEMMAP];
// flag
U8 data_changed = 0;
#define time_base_fast 10 // Time cycle in milliseconds
#define time_base_slow 10000 // Time cycle in milliseconds
#define num_phases 255 // Number of phases
U8 phase_speedy=0, phase_fast=0, phase_slow=0;
unsigned long tmr_fast=0, tmr_slow=0;

En la imagen anterior se muestra la definición de las variables a utilizar, en las cuales se añaden los dos relés incluidos en la tarjeta Xboard Relay en los pines D7 y D8.

Se crea y define una variable con nombre “BOTON” que se encuentra conectada en el pin D6, la cual será usada para cambiar el estado de RELAY7.

Se integra un mapa de memoria que se encuentra por defecto en la carpeta Souliss, y la configuración de tiempos para la ejecución de instrucciones.


Ejemplo_Xboard_Relay | Arduino 1.0.5
Archivo Editar Sketch Herramientas Ayuda
Ejemplo_Xboard_Relay §
void setup()
{
// Configura las direcciones de red
Souliss_SetAddress(network_address_1, network_my_subnet, network_my_supern);
// Carga la dirección IP en el mapa de memoria
Souliss_SetLocalAddress(memory_map, network_address_1);
// Configura el estado para cada variable, T11 es ON/OFF salida digital con opción de timer
Souliss_SetT11(memory_map, RELAY7);
Souliss_SetT11(memory_map, RELAY8);
Souliss_SetT11(memory_map, BOTON);
// Se define si la variable y pin es entrada o salida
pinMode(BOTON, INPUT); // Boton es configurado como entrada
pinMode(RELAY7, OUTPUT); // RELAY7 es configurado como salida
pinMode(RELAY8, OUTPUT); // RELAY8 es configurado como salida
}

En esta imagen se muestra la programación de la estructura de la red IP para Souliss, se configura y se carga en el mapa de memoria anteriormente ingresado.

Se configura para Souliss el estado de cada variable y se cargan en el mapa de memoria.

Se programa como entrada o salida cada variable como lo son BOTON, RELAY7 y RELAY8.

Ejemplo_Xboard_Relay §

```
void loop()
{   // Los métodos Souliss se programan en fases, esto permitirá un equilibrio de carga y el momento adecuado.
  if(abs(millis()-tmr_fast) > time_base_fast)
        {
      tmr_fast = millis();
      phase_fast = (phase_fast + 1) % num_phases;
        // Executa el código cada 3 time_base_fast
        if (!(phase_fast % 3)){
            // Cuando Pin6 sea activado cambia estado de RELAY7
            Souliss_DigIn(6, Souliss_Tln_ToogleCmd, memory_map, RELAY7);
            // Cuando se active desde Souliss cambia estado de RELAY7 y RELAY8
            Souliss_Logic_Tll(memory_map, RELAY7, &data_changed);
            Souliss_Logic_Tll(memory_map, RELAY8, &data_changed);
            // Usar pin 7 Y 8 como salida en RELAY7 Y RELAY8
            Souliss_DigOut(7, Souliss_Tln_Coil, memory_map, RELAY7);
            Souliss_DigOut(8, Souliss_Tln_Coil, memory_map, RELAY8);
            }
            // Executa el código cada 5 time_base_fast
            if (!(phase_fast % 5)){
                // Retreive data from the communication channel
                Souliss_CommunicationData(memory_map, &data_changed);
                }
        }
}
```

Para finalizar se crea el bucle con los cambios de estado para los relés y el uso del botón.

Primero se programa una sentencia If para crear las fases de tiempo que utilizara Souliss para refrescar sus datos y cambiar los dispositivos. Luego se programa otro if dentro del anterior con porcentajes de fase de tiempo de Souliss para lograr un equilibrio en la aplicación y que se cambien correctamente los estados de variables.

A continuación se programa la acción de Boton6, que activara el cambio de estado de RELAY7.

Se crearan la interacción de cambio de estado para la aplicación Souliss. Luego se ingresaran los datos de pin (7 y 8) correspondiente para cada variable (RELAY7 y RELAY8).

Finalmente se carga este sketch en la tarjeta.

CONTROL A TRAVÉS DE APLICACIÓN SOULISS

Terminada la configuración y programación de la tarjeta Xboard Relay, solo hace falta conectarla al computador, conectarla a internet mediante su conector Rj45 e instalas la aplicación Souliss en un teléfono celular con sistema operativo Android. La aplicación es gratuita y se puede descargar directamente desde Play Store.

Con la aplicación ya instala y abierta, se debe abrir la pestaña de opciones y buscar la sección “Opciones de Red”, y luego la opción “Dirección local de Souliss”. Es aquí donde se debe ingresar la IP que se ah configurado en la tarjeta, la cual en este ejemplo es 192.168.0.150. Se selecciona aceptar.

Luego de esto, se retrocede hasta el menú inicial de la aplicación y se selecciona “Escenas”. En este punto la aplicación Souliss nos advertirá de que no existen nodos configurados, y nos da la opción de crearlos para esta configuración y tarjeta. Se selecciona a aceptar y se retrocede nuevamente al menú inicial. Se abre nuevamente “Escenas” en la cual ya están ingresadas 3 opciones, dentro de las cuales está encender luces y apagar luces. Para ejecutar cualquiera de estas instrucciones previamente se debe seleccionar y a continuación se selecciona el icono “Play” en el borde inferior de la aplicación. Con esto se debería ejecutar la opción seleccionara y responder con un encender y o apagar los relés de la tarjeta.

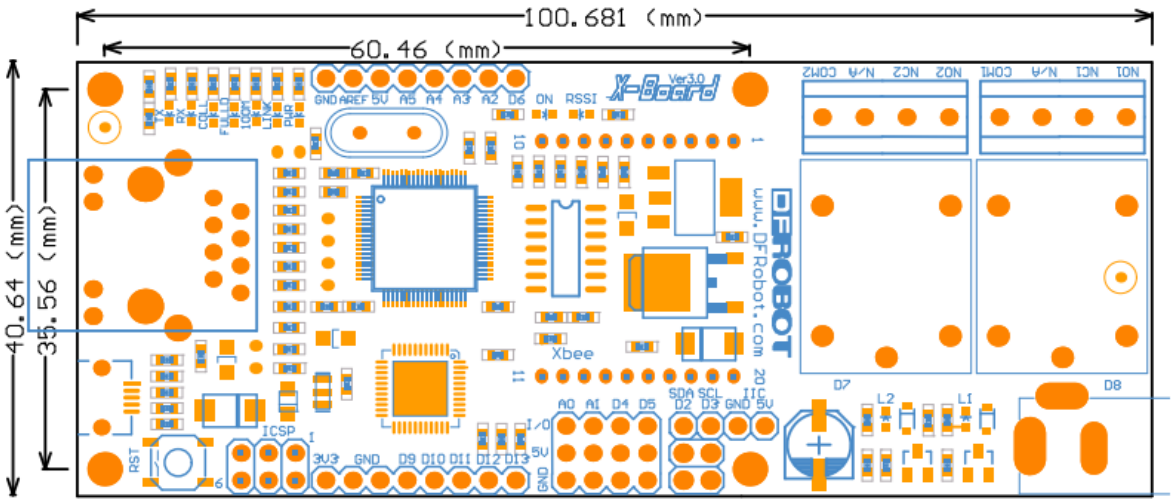
También se puede ejecutar el botón pulsador conectado el cual procede a cambiar el estado del rele7.

CARACTERÍSTICAS ELÉCTRICAS

- *Rangos de voltaje de alimentación: 7,2VDC – 12VDC.*
- *Fuente USB: Micro USB 5V.*
- *Rangos de voltaje de salida: 3,3VDC y 5VDC.*
- *Información Relé:*
  - *Corriente Nominal: 10A (NO) 5A (NC)*
  - *Activación con 5VDC.*
  - *Interfaz Digital*
  - *Señal de control: nivel TTL*
  - *Capacidad de los contactos (carga resistiva): 10A 277VAC 24VDC*
  - *Voltaje Switching Máximo: 250VAC/3VDC 250VAC/30VDC*
  - *Corriente Switching Máxima: 15A*
  - *Potencia Switching Máxima: 2770VA 240W 2770VA 240W*
  - *Calificación UL: 10A 120VAC /10A 277VAC*
  - *Tiempo de operación (a voltaje nominal): 10ms*
  - Tiempo de descarga (a voltaje nominal): 5ms

CARACTERÍSTICAS MECÁNICAS

- *Dimensiones: 40.64mm x 100.681mm.*



PROGRAMA SKETCH ARDUINO

```
#define  BOARDTYPE_INSKETCH
#define  QC_BOARDTYPE  0x30          //Dispositivo a utilizar (Xboard Relay)

#define  GATEWAYTYPE_INSKETCH
#define  QC_GATEWAYTYPE  0x01    //Se activa Gatewaytype (puerta de enlace).

#define  MAC_INSKETCH
const uint8_t MAC_ADDRESS[] = {0x2B, 0xA6, 0x49, 0x6B, 0x00, 0x01};

#include "Souliss.h"     //Se agregan librerías de Souliss
#include "Typicals.h"
#include <SPI.h>

// Direcciones de red

#define network_address_1       0x0096     // 0x0096 = 150 decimal (192.168.0.150)
#define network_my_subnet       0xFF00
#define network_my_supern       0x0000
// Se define nombre de variable y pin al que se conecto.

#define RELAY7  7     // Se crea variable RELAY7 conectada a pin 7
#define RELAY8  8     // Se crea variable RELAY8 conectada a pin 8
#define BOTON   6     // Se crea variable BOTON conectada a pin 6
// Define el mapa de memoria compartida

U8 memory_map[MaCaco_MEMMAP];

// flag
```

```
U8 data_changed = 0;

#define time_base_fast 10              // Time cycle in milliseconds
#define time_base_slow      10000      // Time cycle in milliseconds
#define num_phases          255        // Number of phases

U8 phase_speedy=0, phase_fast=0, phase_slow=0;
unsigned long tmr_fast=0, tmr_slow=0;

void setup()
{
        // Configura las direcciones de red
        Souliss_SetAddress(network_address_1, network_my_subnet,
    network_my_supern);



// Carga la dirección IP en el mapa de memoria
        Souliss_SetLocalAddress(memory_map, network_address_1);


        // Configura el estado para cada variable, T11 es ON/OFF salida digital con
    //opción de timer


        Souliss_SetT11(memory_map, RELAY7);
    Souliss_SetT11(memory_map, RELAY8);
    Souliss_SetT11(memory_map, BOTON);


        // Se define si la variable y pin es entrada o salida
        pinMode(BOTON, INPUT);          // BOTÓN es configurado como entrada
    pinMode(RELAY7, OUTPUT);    // RELAY7 es configurado como salida
    pinMode(RELAY8, OUTPUT);    // RELAY8 es configurado como salida
}
```

```
void loop()
{
        // Los métodos Souliss se programan en fases, esto permitirá un equilibrio de      //carga y
el momento adecuado.

        if(abs(millis()-tmr_fast) > time_base_fast)
        {
                tmr_fast = millis();
                phase_fast = (phase_fast + 1) % num_phases;


                // Executa el código cada 3 time_base_fast
                if (!(phase_fast % 3))
                {
                        // Cuando Pin6 sea activado cambia estado de RELAY7
                        Souliss_DigIn(6, Souliss_T1n_ToogleCmd, memory_map, RELAY7);


                        // Cuando se active desde Souliss cambia estado de RELAY7
        // Cuando se active desde Souliss cambia estado de RELAY8
                        Souliss_Logic_T11(memory_map, RELAY7, &data_changed);
        Souliss_Logic_T11(memory_map, RELAY8, &data_changed);

                        // Usar pin 7 como salida en RELAY7
                        // Usar pin 8 como salida en RELAY8
                        Souliss_DigOut(7, Souliss_T1n_Coil, memory_map, RELAY7);
        Souliss_DigOut(8, Souliss_T1n_Coil, memory_map, RELAY8);
                }
```

```
// Executa el código cada 5 time_base_fast
        if (!(phase_fast % 5))
        {
            // Retreive data from the communication channel
            Souliss_CommunicationData(memory_map, &data_changed);
            }
        }
```

HISTORIA DEL DOCUMENTO

Revisión	Fecha	Editado por	Descripción/Cambios
1.0	19 de febrero de 2014	Patricio Peña M.	Versión inicial del documento